

周报 2019.4.29~5.5

Done

1.React-Redux学习

- 一些基本概念：

第二阶段

状态提升

- 并列的子组件无法直接共享状态，需要最近公共父节点通过props来传递
- 规则：某个状态受到多个组件依赖时，就执行状态提升
- 出现的问题：无限制的提升？层层编写函数？ -> 解决方案：Redux

组件挂载

- 挂载：初始化组件，render渲染JS对象，构造DOM元素，插入页面
- 具体的函数和执行顺序：
 - constructor：初始化组件
 - componentWillMount：组件挂载开始之前，也就是在组件调用 render 方法之前调用。
 - render：组件渲染，生成JavaScript对象
 - componentDidMount：组件挂载完成以后，也就是 DOM 元素已经插入页面后调用。
 - componentWillUnmount：组件对应的 DOM 元素从页面中删除之前调用。

组件挂载的作用

- componentWillMount：Ajax数据拉取、定时器的启动
- componentDidMount：动画的启动
- componentWillUnmount：定时器的清理

SetState更新组件

- shouldComponentUpdate(nextProps, nextState)：你可以通过这个方法控制组件是否重新渲染。如果返回 false 组件就不会重新渲染。这个生命周期在 React.js 性能优化上非常有用。
- componentWillReceiveProps(nextProps)：组件从父组件接收到新的 props 之前调用。
- componentWillUpdate()：组件开始重新渲染之前调用。
- componentDidUpdate()：组件重新渲染并且把更改变更到真实的 DOM 以后调用。

DOM操作

- ref属性：获取DOM元素来调用DOM的API（尽量避免使用，react有自动更新和事件

监听)

容器类组件

- 组件标签内的HTML结构，可以通过props.children获取数组，安置JSX元素

第三阶段

高阶组件

- 一种函数，输入一个组件，加上自己的功能，输出新的组件。用于复用多个类似组件的相同操作

Context

- 全局变量，公共，用getChildContext设置，使得所有子组件都可以访问this.context（一般不用）

模拟Redux

- 问题：组件之间要共享数据+数据被任意修改 之间的矛盾
- 方法：
 - react采用dispatch函数，限制修改共享数据appstate的方法
 - 函数createStore，参数state表示数据，stateChanger表示修改数据的方法；返回的对象是getState获取数据，dispatch修改数据的实例，subscribe监听数据变化情况
- 问题：每对数据appstate执行一次修改，就会有多余的渲染
- 方法：
 - 产生数据修改时，新建一个对象，把其中不变的数据浅复制过来，把变化的数据修改。然后，当react执行渲染和DOM操作时，newState和oldState比较，不变的就不再渲染
- reducer：
 - 纯函数function reducer(state, action)，输入你的数据，和数据修改方式；返回一个初始化的数据或者，创建一个新的数据对象
 - const store = createStore(Reducer) 返回return { getState, dispatch, subscribe }
- 总结：

我们从一个简单的例子的代码中发现了共享的状态如果可以任意修改的话，那么程序的行为将非常不可预料，所以我们提高了修改数据的门槛：你必须通过 `dispatch` 执行某些允许的修改操作，而且必须大张旗鼓的在 `action` 里面声明。

这种模式挺好用的，我们就把它抽象出来一个 `createStore`，它可以产生 `store`，里面包含 `getState` 和 `dispatch` 函数，方便我们使用。

后来发现每次修改数据都需要手动重新渲染非常麻烦，我们希望自动重新渲染视图。

- 。所以后来加入了订阅者模式，可以通过 `store.subscribe` 订阅数据修改事件，每次数据更新的时候自动重新渲染视图。

接下来我们发现了原来的“重新渲染视图”有比较严重的性能问题，我们引入了“共享结构的对象”来帮我们解决问题，这样就可以在每个渲染函数的开头进行简单的判断避免没有被修改过的数据重新渲染。

我们优化了 `stateChanger` 为 `reducer`，定义了 `reducer` 只能是纯函数，功能就是负责初始 `state`，和根据 `state` 和 `action` 计算具有共享结构的新的 `state`。

模拟React-Redux

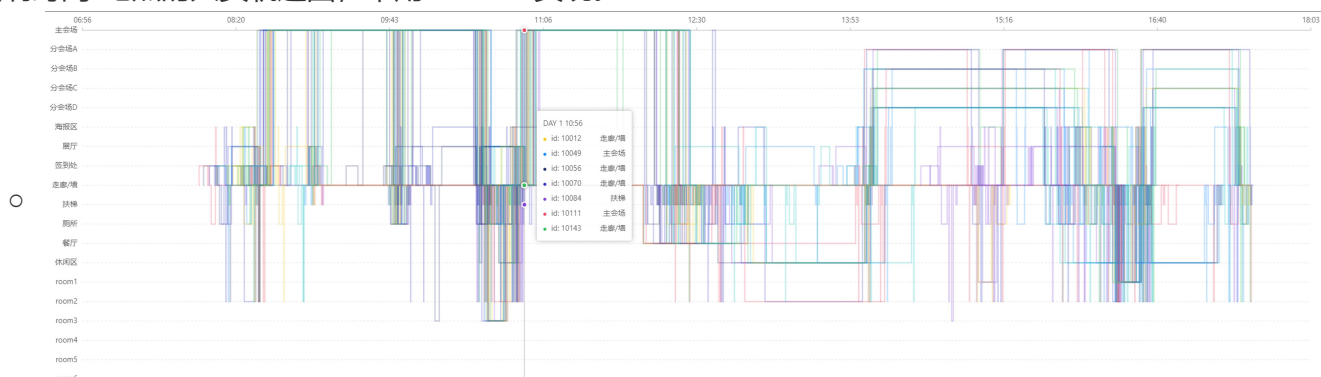
- 。Dumb component只依赖传入参数props和自己的state，复用性很强
- 。把使用全局变量context的语句清除，改用一个高阶组件函数connect来和context沟通，用provider组件作为根节点存放context

真正的React-Redux

- 。smart组件处理特定的应用逻辑，不考虑复用；dumb组件用于复用，减少对其他组件的依赖
- 。所有的 Dumb 组件都放在 components/ 目录下，所有的 Smart 的组件都放在 containers/ 目录下

2.ChinaVis 2019 挑战赛 项目

- 。绘制时间-地点的人员轨迹图，采用Bizchart实现。



小结

本周学习了react.js，上手了挑战赛的项目。虽然说好五一节要学习，但是也就学习了一天。不过项目画图已经在有条不紊的开展中了。

Plan

短期计划

1. 看联邦学习、迁移学习的实现方法和样例
2. 完善ChinaVis2019的视图
3. 组会报告可以开始写了
4. 毕业设计可以开始写了

中期计划

1. 完成本科毕业设计
2. 完成组会报告

长期计划

1. 6月份时，完成上述几个事项